DATAMATION® 65

April

SOFTWARE FOR RANDOM ACCESS PROCESSING

SOFTWARE FOR RANDOM ACCESS PROCESSING

the parts list application

by CHARLES W. BACHMAN

The Integrated Data Store[1] is being applied to an increasing number and variety of data processing tasks. One of the earliest was the processing of engineering parts lists. Suited to this assignment by reason of its ability to handle the most complex information structures, the Integrated Data Store is a programming system designed to organize and process records within a random access data storage device, such as a disc, drum or loops. It provides a means of defining data structures (records and chains) and a set of procedural verbs (STORE, RETRIEVE, MODIFY AND DELETE) that operate within the defined data structure.

The facilities of the Integrated Data Structure are supplementary to those of standard programming languages such as COBOL and FORTRAN. Jointly they provide a new opportunity to organize and process data to achieve specified business objectives.

The principal data organization mechanism used by the Integrated Data Store is the chain, which permits a group of related records to be tied together. This tie is accomplished by storing, within each record of the group, the address of the next record of the group. Thus a chain becomes a circular association of records. Fig. 1 illustrates a sample chain. The small diagram at the lower right is the shorthand description of the same chain.

Each chain contains one and only one master record. This record is specified in the data definition as the chain's master. Whenever a master record is stored, the chain is created and the master record's own address is stored in its pointer field.

Each chain may contain any of several detail records —zero, one, 10, or 10,000. Whenever a detail record is stored, it is automatically linked into the chains within which it is defined as a detail. The prior record in the chain

[1]Bachman, C. W.; Williams, S. B., "A General Purpose Programming System for Random Access Memories," *Proceedings*, Fall Joint Computer Conference, 1964.

Mr. Bachman is manager of Software Planning for GE's Computer Dept., Phoenix, Ariz., and is the inventor of the I-D-S technique. While with Dow Chemical, he chaired the SHARE committee that initiated development of the 9PAC system. He joined corporate GE in '60, specializing in integrated systems R&D. He holds a BS in mechanical engineering from Michigan State U., and an MS in engineering and business from Pennsylvania.

Reprinted from DATAMATION, April, 1965, courtesy of F.D. Thompson Publications, Inc.

is modified to store the address of the new detail record in its pointer field. The new detail will contain the address of the next record of the chain that had been stored in the prior record.

A new detail will be inserted into its chain according to the ordering rule specified in the data description. In the product structure example to be given, one chain will be ordered in data value sequence. The other chain will be ordered by inserting each new detail record as the first detail in the chain.

engineering parts list

The engineering parts list, or bill of materials as it is frequently known, is the foundation of a manufacturing business. Purchase orders, sales invoices, time cards, and stock tickets are essential; but they all reflect peripheral actions. The product must still be made so that it can be sold.

The parts list and the listed engineering drawings are the original Engineering definition of what the product is. It becomes the Manufacturing description of what has to be used to make the product. It is the Accounting basis for preparing cost studies. Finally, it is the customer's basis for ordering spare parts during the life of the product.

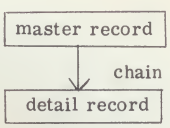
At first glance (Fig. 2) the information content of a parts list is deceptively simple. A single parts list identifies and describes a product. It then lists serially the parts or subassemblies used to produce the product, and indicates the quantity required of each part. Normally each of the items listed on a parts list is also given an item number that is relevant to that particular parts list - - i.e. item 1, item 2, item 3, etc.

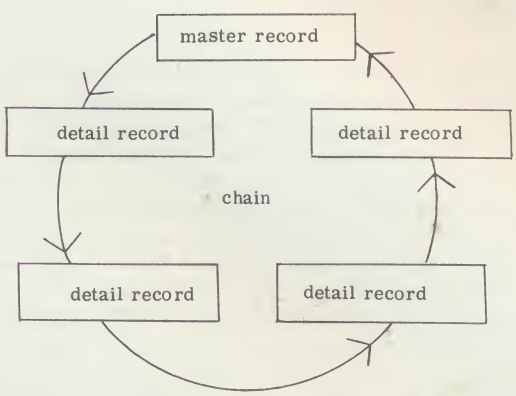
A very simple data structure can be used to transfer a parts list onto a computer. A master record can be placed on a punched card or magnetic tape representing the parts list heading and a detailed or trailer record added following the master for each item on the parts list. This is the usual approach taken by file designers in processing this type of engineering information. The serial nature of punched cards or magnetic tape closely approximates the serial geometry of paper, the original form of data processing.

This solution suffers from the same problem that affects the original paper parts list, which in itself was an information processing abstraction of the original product structure that it describes. The original product is a thing made out of things which, in turn, are made of other things. Frequently it requires many hundreds of interlocking parts lists to completely describe an item such as a computer or disc storage unit that a customer would want to buy. This is the product structure which really needs to be represented within a computerized information system. From this structure, a simple parts list can be reconstructed and printed. A report showing every place a part is used (where-used list) can be printed. The effect of producing a given quantity of an end product can be reflected down to the lowest level in the product structure through a parts explosion procedure.

Random access data storage units, such as disc files drums, and magnetic loops, provide an opportunity to build and process information structures of more than one dimension. Given this ability, a fresh new approach may be taken in organizing product structure. A systematic review of the information content of a parts structure is required.

Fig. 1 Sample Chain Structure





Note: The arrow is schematically representing the fact that each record contains the address of the next record in the chain.

kinds of parts information needed

Fig. 3 (p. 38) shows a schematic representation of the product structure built from the parts lists of Fig. 2. In this illustration seven material items and seven relationships between pairs of material items are shown. Boxes represent material items, lines designate the quantity relationship. Numerals with the relationship lines denote quantities of each respective item required for the manufacture of other items.

For example, the diagram shows that Material Item B requires two of Material Item D, 12 of material Item R, and three of M. In normal manufacturing practice, any given material item may require quantities of from one to 1,000 other material items for its assembly. Of course, if the material item is purchased completely assembled, it may not require any other material item for its manufacture. Material items F, G, and T are examples.

Fig. 2 Sample Parts Lists

item B

item no	quantity	mat'l. item
1	2	D
2	12	R
3	3	M

item D

item no	quantity	mat'l. item
1	1	F
2	2	G
3	6	T

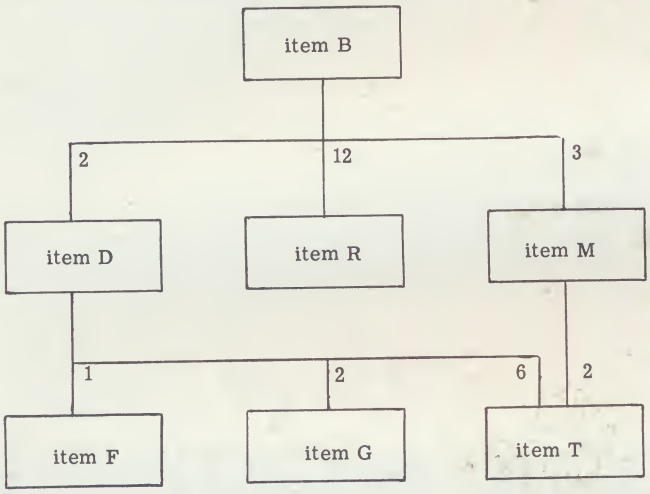
item M

item no	quantity	mat'l. item
1	2	T

In the discussion of material items, it should be noted that the term *higher-level* is applied to a material item which is being produced in the manufacturing operation; *lower-level* is the term for one being consumed in the manufacturing process. The former might be the drive mechanism of a tape transport; the latter a spindle or motor used in its manufacture.

Any given item may be required for use in one or more higher assemblies. Item T, for example, is used by both

Fig. 3 Product Structure



Items D and M. Another item may have no requirements; Item B would be such a piece.

In an actual manufacturing situation, 10,000 material items are frequently found to exist, and 40,000 items are not unusual. The total number of quantity relationships will average from one to three times the number of material items.

From the foregoing discussion of product structuring, it may be seen that: (1) Two different units of information are involved in a parts structure—namely, material items and quantity relationships; (2) Quantity relationships between the material items create a complex network describing the assembly process; and (3) These networks are typically very large in size, comprising tens of thousands of material items and relationships.

Two records will be designed to store information about the parts list structure. The first is called a *material item record.* This will contain two data fields: the *material identification* and the *material description.* The second record is the *submaterial* record. This represents a quantity relationship between two material item records. It contains two data fields, *quantity required,* and *item number.* The *quantity required* field is self-explanatory. The *item number* field is used to number the several submaterial records relating to a given parts list. These numbers, when combined with the material identification of the higher-level material record, serve uniquely to identify each submaterial record.

Two chains have been established to control the association of these records. The first, known as the "call-out" chain (Fig. 4), has as its master record the material item record. This chain will contain as many submaterial records as required to specify all the quantities of lower level items necessary to manufacture the material item designated by the master of the chain.

It should be noted that as each material record is stored, it is automatically made the master record of a call-out chain. Of course, no detail records would be in the chain at that time. Still, there would be a chain field in the record containing the address of the next record in the chain. This next record address would be the address of the material item record itself.

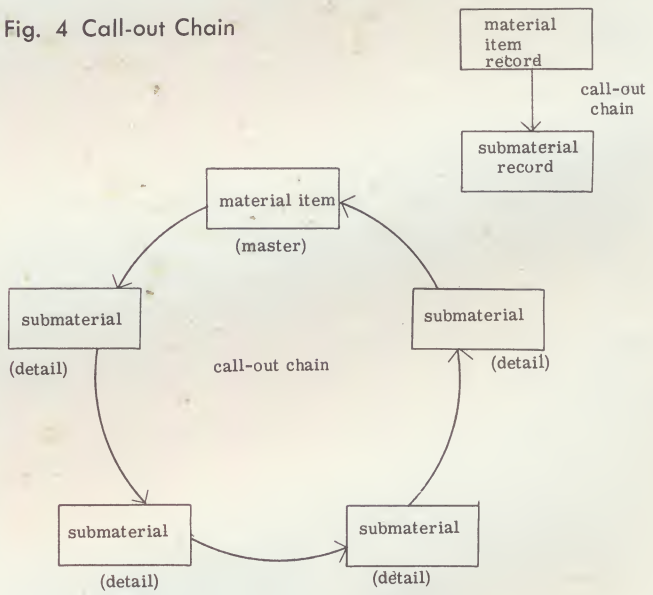
Whenever a new submaterial record is to be stored, it must select one material item record from the thousands of existing material item records to be its appropriate master. It will select the material item record specified by the material identification number which was supplied in the input along with the item number and the quantity required. If for some reason the material item record cannot be retrieved, then an error is indicated and the storage process terminated.

Once the correct material item record is retrieved, it is necessary to determine whether a submaterial record with the same item number is already in the call-out chain. If a duplicate record is detected, the storage process is terminated and an error noted. If the new record is not a duplicate, it will be linked into its proper place in the chain, based upon the sequence of item numbers. The call-out chain is a data-sequenced chain.

A second chain, the "where-used" chain, defines all of the relationships for a material item record where that material item is used in the manufacture of a higher-level material item. The material item record is the master record of this chain too. The submaterial record is again the detail in this chain. In this way the submaterial record creates, via the two chains, the desired relationship between two different material item records.

The small diagram in the upper right corner of Fig. 7 is the record structure diagram. It specifies that there

Fig. 4 Call-out Chain



are two record types (the material item record and submaterial record) and two chains relating them (the call-out chain and the where-used chain). The direction of the arrows indicates that the material item record is the master record of both chains. However, it should be noted that a particular submaterial record always has two different material item records (same record type) as the master records for its where-used and call-out chains.

The submaterial record selects its master record (material item record) in the where-used chain, based upon a second material identification known as submaterial identification. The submaterial identification field is the fourth field supplied with the input. There is no need to sequence the where-used chain; therefore each new submaterial record is inserted as the first detail record in the

where-used chain at the time of storage. This form of insertion uses the least amount of computer time.

Fig. 5 shows a sheet of I-D-S/COBOL data definitions describing the material item and submaterial item records. Fig. 6 is a companion record layout sheet showing what is stored in each character of each record. These data definitions generally follow the COBOL data descriptions, with the addition of certain Integrated Data Store statements necessary to guide the automatic storage and retrieval functions.

The material item record is specified as a calculated record by the statement, "RETRIEVAL VIA CALC CHAIN." The statement, "RANDOMIZE ON MATERIAL-ID" identifies the data field that will be operated upon by a randomizing routine to derive the location where a given material item record would be normally stored. The Calc Chain is a special chain designed to link together all of the records that randomize to the same location. Thus a scan of all the records in a particular Calc Chain would give access to all the records that have randomized to a location, even though an overflow situation prevented the record from being stored in its normal place.

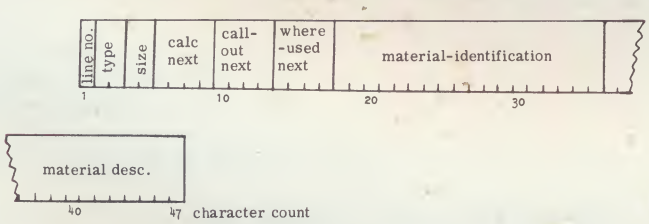
The data "page" is the basic addressable unit of mass memory, which is the location where a record is stored. Only complete pages are transmitted between the computer and its mass memory. Each record's address consists of its page number plus the line number assigned to the record within that page. The data description in Fig. 5 specifies a particular file with 8,192 pages of storage capacity with each page large enough to store 2,880 characters of data. The size of the page and number of pages are specified by the file designer to gain the best performance for the specific mass memory device he is employing.

Fig. 5 I-D-S/COBOL Data Definitions

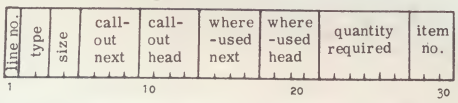
```
DATA DIVISION
IDS SECTION
MD MATERIAL-FILE; PAGE CONTAINS 2880 CHAR.; FILE CONTAINS 8192 PAGES.

01 MATERIAL-ITEM; TYPE IS 100; RETRIEVAL VIA CALC CHAIN.

02 MATERIAL-ID; CLASS 18 ALPHANUMERIC.
02 MATERIAL-DESC; CLASS 12 ALPHANUMERIC.

98 CALC CHAIN DETAIL; RANDOMIZE ON MATERIAL-ID.
98 CALL-OUT CHAIN MASTER.
98 WHERE-USED CHAIN MASTER.


01 SUB-MATERIAL; TYPE IS 200; RETRIEVAL VIA CALL-OUT CHAIN.

02 ITEM-NO; CLASS 3 NUMERIC.
02 QUANTITY-REQUIRED; CLASS 6 COMPUTATIONAL-N.

98 CALL-OUT CHAIN DETAIN; SELECT UNIQUE MASTER; MATCH-KEY IS
   MATERIAL-ID; CHAIN-ORDER IS SORTED; DUPLICATES NOT ALLOWED;
   ASCENDING SORT-KEY IS ITEM-NO; LINKED TO MASTER.
98 WHERE-USED CHAIN DETAIL; SELECT UNIQUE MASTER; MATCH-KEY IS
   SUB-MATERIAL-ID; SYNONYM SUB-MATERIAL-ID EQUALS MATERIAL-ID:
   CHAIN-ORDER IS FIRST; LINKED TO MASTER.
```

Fig. 6 I-D-S File Companion Record Layout

Material-item Record



Submaterial Record

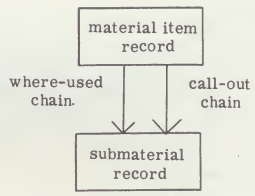


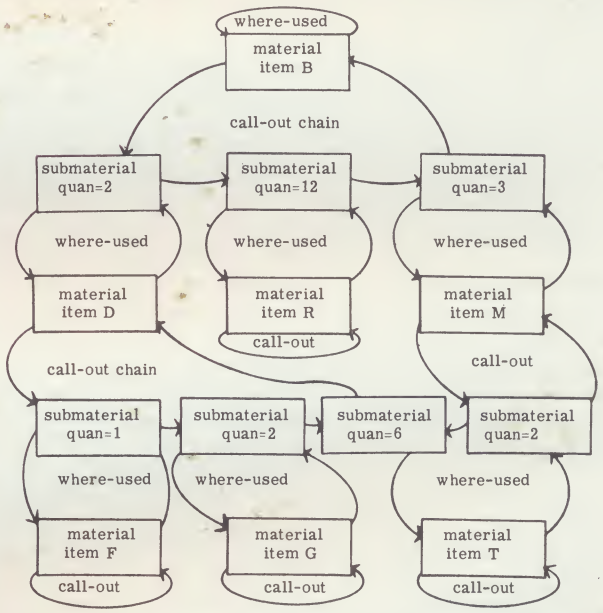
The Integrated Data Store will automatically attempt to store a new submaterial record in the same page as its master record in the call-chain. In normal operation, most of the submaterial records will be stored in the same data page as the material item record, or in the adjacent one. Therefore, no disc time, or only latency time, is involved in retrieving all the records in the call-out chain for a particular material item record. The desire to store a submaterial record near its call-out chain master is specified by the statement in the data description, "RETRIEVAL VIA CALL-OUT CHAIN."

The level 98 entries in the data division specify the relationship of a record to a chain. The initial clause of a level 98 entry names the chain and indicates whether the record is associated with the chain as a master or a detail. In the case of a detail association, the rules for selecting the particular chain master and inserting record into the chain are also specified.

The series of parts lists that made up Fig. 2 were translated into the product structure shown in Fig. 3. This same product structure, when expressed through the Integrated Data Store as records and chains, would appear as the information structure of Fig. 7. A material item record is stored for each material item. A submaterial record is stored for each quantity relationship expressed. Finally, the call-out chains and where-used chains tie together the material item and submaterial records to achieve the information structure which exactly models the original product structure.

Fig. 7 Information of Figs. 1 and 2, organized into material item records and submaterial records, and chained together to create a product structure.





the file-building routine

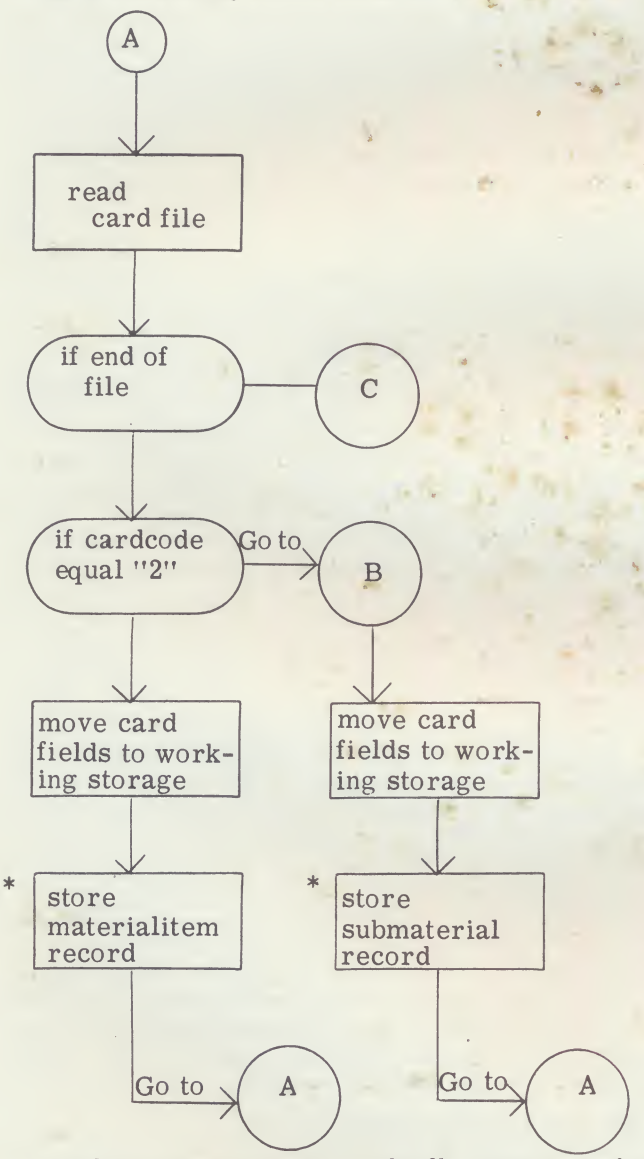
The building of a file such as the one expressed by the data description of Fig. 5 and the actual records and chains of Fig. 7 would be straightforward. Two different input cards would be designed. The first one would contain a material identification and material description. Its function would be to create a new material item record. For control purposes, this card will contain a card code of "1." A second card would contain the information

required to store a new submaterial record. Its card code would be "2."

Fig. 8 is the flow chart of the file-building routine that would read these input cards and store the indicated material item records and submaterial records. It should be noted that all of the functions of finding space to store a new record, initializing chains for master records, and inserting detail records in chains is automatically carried out during the execution of a STORE command. The particular action to be taken is dependent upon the data description of the record to be stored.

In a comprehensive file maintenance routine, additional input card formats would be established for the specification of record modification and deletion functions. I-D-S statements such as "RETRIEVE SUBMATERIAL RECORD, REPLACE QUANTITY-REQUIRED FIELD" and "RETRIEVE MATERIAL ITEM, DELETE" would be used to direct the Integrated Data Store to carry out the file maintenance functions required.

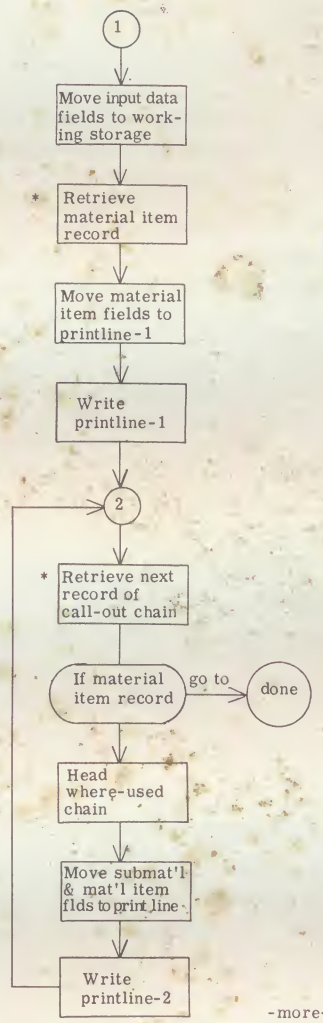
Fig. 8 File Building Routine Flowchart



It will be recalled that the purpose of building the file in the first place was to permit the subsequent production of parts lists, where-used lists and parts explosions. A flow chart of the steps necessary to accomplish the preparation of a parts list report is shown in Fig. 9.

A parts list report includes an initial print line (printline-1) containing the material identification and description of the material item for which the report is being prepared. It also includes as many detail print lines (printline-2) as there are items on the parts list. There will be one detail print line for each submaterial record which is a detail in the call-out chain. This detail line contains the item number, quantity required, material identification, and description of the lower-level material item. The item number and quantity required come from the submaterial record. The material identification and the description of the lower-level material item come from the lower-level material item record at the same time the submaterial record is retrieved, through the use of the clause, "HEAD WHERE-USED CHAIN."

Fig. 9 Flowchart of Steps Necessary to Prepare a Parts List Report



-more-

*Integrated Data Store Functions

This extension of the RETRIEVE command will automatically trace the detail records of the specified chain and retrieve the master record and move its data content to working storage. This action can be accelerated by inserting the clause "LINKED TO MASTER" in the level 98 chain definition of the Data Division. Use of this clause will cause each detail in that chain to contain an additional field. The address of the master record of the chain will be inserted in this additional chain field when the detail is linked into the chain. If this extra pointer is included, the HEAD function can retrieve the master record directly without tracing through all the detail records of the chain.

Note that the HEAD clause is a convenient means for extending retrieval functions to automatically retrieve the

Fig. 10 I-D-S/COBOL Statements for Compiling Coding to Produce Parts List Based on Flowchart in Fig. 9

```
STEP-1.
      MOVE CARD-ID TO MATERIAL-ID, PRINT-ID.
*     RETRIEVE MATERIAL-ITEM RECORD; MOVE TO WORKING-STORAGE;
          IF ERROR GO TO STEP-4.
      MOVE MATERIAL-DESC TO PRINT-DESC.
      WRITE PRINT-LINE-1.
STEP-2.
*     RETRIEVE NEXT RECORD OF CALL-OUT CHAIN;
          IF ERROR GO TO STEP-4;
          OTHERWISE IF MATERIAL-ITEM RECORD GO TO STEP-3;
          OTHERWISE IF SUB-MATERIAL MOVE TO WORKING-STORAGE:
          HEAD WHERE-USED CHAIN.
      MOVE ITEM-NO TO PRINT-ITEM-NO.
      MOVE QUANTITY-REQUIRED TO PRINT-QUANTITY.
      MOVE MATERIAL-ID TO PRINT-SUB-ID.
      MOVE MATERIAL-DESC TO PRINT-SUB-DESC.
      WRITE PRINT-LINE-2.
      GO TO STEP-2.
STEP-3.
      STOP RUN.
* Integrated Data Store Statements
```

information in the master when required. This information is, in truth, part of the detail record; however, to eliminate redundancy and simplify file maintenance, it is stored in the master record and shared with the other detail records in the chain.

Fig. 10 gives the I-D-S/COBOL procedure division statements which would cause the necessary coding to be compiled to produce a parts list. Asterisks indicate I-D-S commands. The other nine commands are standard COBOL.

The coding to produce a where-used report is similar to that of the parts list report, with the exception of the "RETRIEVE NEXT" command. To produce a where-used report, this command would be changed to read, "RETRIEVE NEXT RECORD OF WHERE-USED CHAIN; . . . HEAD CALL-OUT CHAIN."

product explosion reports

The preparation of a parts explosion report is an important function of a material structure file. It is somewhat more complicated to produce than a parts list and where-used list. However, the I-D-S ability to traverse chains linking the information structure facilitates the preparation of product explosion reports. The myriad pathways of the parts structure make it necessary to do some trail-blazing to avoid losing the route of exploration. An array named "LEVEL-REFERENCE" is used to store the addresses of successive submaterial records as the routine probes downward through the parts structure. The address is used when the bottom level is reached, to back up to the last book and to probe the next branch of the tree-like structure. The array index is a field named "LEVEL." A second array, "LEVEL-QUANTITY," keeps track of the exploded quantity at each book in the structure. Note it uses the same index, "LEVEL." These two arrays need be only as long as the depth of the product structure. The very simple example we have shown has a depth or level of three. Complicated products may have a depth or level of 10 to 15.

Referring back to Fig. 7, the process of explosion would be down from material Item B through the quantity relationship of "2" to Item D, through the quantity relationship "1" to material Item F, then to material Items G, T and R until complete. Fig. 11 illustrates the parts explosion report, based on a requirement for only one material Item (B).

Fig. 11 Simple Parts Explosion Report

Level	Quantity	Description
1	1	item B
2	2	item D
3	2	item F
3	4	item G
3	12	item T
2	12	item R
2	3	item M
3	6	item T

Note that material Item T appears twice in the report because it is used in the manufacture of both material Items D and M. If desired, a slightly more complicated procedure would permit the accumulation of the quantity exploded for these multiple-occurrence items. Programs exist to do level-by-level material requirements explosions as well as cost implosions on the same file.[2]

Fig. 12 lists I-D-S/COBOL procedure statements for carrying out a report preparation for a parts explosion procedure. I-D-S commands are indicated by asterisks. Note that four of the seven I-D-S retrieved procedures are used in this example: "RETRIEVE;" "RETRIEVE NEXT RECORD OF data-name CHAIN;" "RETRIEVE MASTER OF data-name CHAIN," and "RETRIEVE DIRECT."

The field, "DIRECT-REFERENCE" is a communication cell used by the Integrated Data Store. After the execution of each I-D-S statement the address of the record processed is left in this cell. The RETRIEVE DIRECT verb picks up the address of the record to be retrieved in this same cell. A record address as used in this context, and in the context of chain pointers, is made up of the record's page number and line number. These are not hardware addresses.

Fig. 12 I-D-S/COBOL Procedure Statements

```
STEP5.
    MØVE CARD-ID TØ MATERIAL-ID.
*   RETRIEVE MATERIAL-ITEM RECØRD, MØVE TØ WØRKING-STØRAGE;
        IF ERROR GØ TØ STEP4.
STEP6.
    MULTIPLY QUANTITY-REQ BY LEVEL-QUAN (LEVEL) GIVING EXPLØDED-QUAN.
    ADD ØNE TØ LEVEL.
    MØVE EXPLØDED-QUAN TØ LEVEL-QUAN (LEVEL), PRINT-QUAN.
    MØVE MATERIAL-ID TØ PRINT-ID.
    WRITE PRINT-LINE-3.
STEP7.
*   RETRIEVE NEXT RECØRD ØF CALL-ØUT CHAIN;
        IF ERRØR GØ TØ STEP4;
        ØTHERWISE IF MATERIAL-ITEM RECØRD GØ TØ STEP8;
        ØTHERWISE IF SUB-MATERIAL RECØRD MØVE TØ WØRKING-STØRAGE.
    MØVE DIRECT-REFERENCE TØ LEVEL-REF (LEVEL).
*   RETRIEVE MASTER RECØRD ØF WHERE-USED CHAIN;
        MØVE TØ WØRKING-STØRAGE;
        IF ERRØR GØ TØ STEP4.
    GØ TØ STEP6.
STEP8.
    SUBTRACT ØNE FRØM LEVEL.
    IF LEVEL EQUAL ZERØ GØ TØ STEP9.
    MØVE LEVEL-REF (LEVEL) TØ DIRECT-REFERENCE.
*   RETRIEVE DIRECT;
        IF SUB-MATERIAL RECØRD GØ TØ STEP7.
    GØ TØ STEP7.
STEP9.
    STØP RUN.
* Integrated Data Store Statements
```

conclusion

Experience to date using the Integrated Data Store shows that its ability to eliminate redundant data can save from 25 to 40% of the space required by conventional file designs. As an example, a material identification number which may be as many as 18 characters in length, need appear only once in the entire file. Normally it would appear in the file once for every time that it appeared on a parts list.

The I-D-S data structuring ability and powerful procedural verbs (STORE, RETRIEVE, MODIFY, AND DELETE) automatically carry out record processing functions which are lengthy, complicated and difficult to program and debug using normal disc file programming techniques. Reductions in time and expense of 25% have already been realized by using the Integrated Data Store to program large integrated business systems.

Operating times have been reduced up to 50% because efficient buffering techniques, data blocking and good data organization are uniformly applied to all I-D-S programs. Implementation schedules and the supply of top-notch programmers are usually too tight to permit hand-coded programs to reach or maintain their potential effectiveness.

The Integrated Data Store received its first thorough testing by General Electric in the summer of 1963 using existing disc storage unit parts explosion routines as test controls. These tests were run on GE-225 computers. Today there are four large integrated business systems pioneering the use of the Integrated Data Store. Several of these systems process their product structure in essentially the form indicated in this article. All of these systems are on GE-215, 225 or 235 computers.

The General Electric Computer Dept. is currently implementing Integrated Data Store Systems for the GE-400 series and 600 series computers. These systems will operate in conjunction with the GE COBOL processors and will be available this coming summer. ■

[2]General Electric Computer Dept.'s publication CPB-279P, "Disc Storage Access Parts Explosion," documents other means of carrying out parts explosions.

OFFICES

ATLANTA, GEORGIA
BOSTON, MASSACHUSETTS
CHARLOTTE, NORTH CAROLINA
CHICAGO, ILLINOIS •
CINCINNATI, OHIO
CLEVELAND, OHIO •
COLUMBUS, OHIO
DALLAS, TEXAS •
DENVER, COLORADO
DES MOINES, IOWA
DETROIT, MICHIGAN
HONOLULU, HAWAII
HOUSTON, TEXAS
HUNTSVILLE, ALABAMA
INDIANAPOLIS, INDIANA
JACKSONVILLE, FLORIDA
KANSAS CITY, MISSOURI
LOS ANGELES, CALIFORNIA
LOUISVILLE, KENTUCKY
MEMPHIS, TENNESSEE
MINNEAPOLIS, MINNESOTA
NEW ORLEANS, LOUISIANA
NEW YORK, NEW YORK •
OKLAHOMA CITY, OKLAHOMA
ORLANDO, FLORIDA
PHILADELPHIA, PENNSYLVANIA
PHOENIX, ARIZONA •
PITTSBURGH, PENNSYLVANIA
PROVIDENCE, RHODE ISLAND
SALT LAKE CITY, UTAH
SAN FRANCISCO, CALIFORNIA
SCHENECTADY, NEW YORK •
SEATTLE, WASHINGTON
ST. LOUIS, MISSOURI
SYRACUSE, NEW YORK
TALLAHASSEE, FLORIDA
WASHINGTON, D.C. AREA •

Africa:
Bull-General Electric and Affiliates
Abidjan, Algiers, Casablanca, Dakar, Tananarive

Australia:
Australian General Electric Pty., Ltd.
Melbourne, • Sydney •

Canada:
Canadian General Electric Co., Ltd.
Montreal, Toronto

Europe:
Bull-General Electric and Affiliates
Amsterdam, Athens, Basel, Belgrade, Bern, Brussels, Cologne, Copenhagen, Geneva, Helsinki, Lisbon, London, Madrid, Oslo, Paris, Stockholm, Vienna
Olivetti-General Electric
Bologna, • Milan, • Rome, Turin

Orient:
Bull-General Electric and Affiliates
Beirut, Istanbul, Tokyo

South America:
Bull-General Electric and Affiliates
Buenos Aires, Mexico, D.F., Montevideo, São Paulo

or write Drawer 270, Phoenix 1, Arizona

• Information Processing Centers in these cities offer complete computer services.

Progress Is Our Most Important Product

GENERAL ELECTRIC

COMPUTER DEPARTMENT

CPBR-438 LITHO IN U.S.A.